# Further SRM v2.2 Proposed Changes
# SRM_v2.2_draft_L8

Compiled by Alex Sim at LBNL

Inclusive of SRM v2.2 draft 3 by
Timur Perelmutov at FNAL

With contributions from
Jean-Philippe Baud and Maarten Litmaath at CERN
and Arie Shoshani at LBNL

May 11, 2006

**Abstract**

In regards to recent discussion with WLCG Data Management Coordination Group for the requirements of the LHC experiments of the Grid Storage Interfaces, we propose the changes to SRM v2.1.2 specification.
This proposed SRM version 2.2 changes extend and update the current SRM v2.1.2 specification. When generally accepted, missing detailed behaviors and defaults will be worked out, and the final changes will be consolidated into the SRM specification to be a part of SRM v2.2 specification.

Notes

- Red highlighted texts indicate that those will be removed from the current v2.1.2 specification.
- Yellow highlighted texts indicate that those will be added to the current v2.1.2 specification.

**Proposed V2.2 Changes**

enum TSpaceType {Volatile, Durable, Permanent}

enum **TRetentionPolicy** { REPLICA , OUTPUT , CUSTODIAL }

- Quality of Retention (Storage Class) is a kind of Quality of Service. It refers to the probability that the storage system lose a file. Numeric probabilities are self-assigned.
    - Replica quality has the highest probablity of loss, but is appropriate for data that can be replace because other copies can be accessed in a timely fasion.

- Output quality is an intermediate level and refers to the data which can be replace by lenghty or effort-full processes.
- Custodial quality provides low probability of loss.

enum **TAccessLatency** { ONLINE, NEARLINE }

- Files may be Online, Nearline or Offline. These terms are used to describe how latency to access a file is improvable. Latency is improved by storage systems replicating a file such that its access latency is online.
  - The ONLINE cache of a storage system is the part of the storage system which provides file with online latencies.
  - For completeness, we also describe OFFLINE here.
  - ONLINE has the lowest latency possible. No further latency improvements are applied to online files.
  - NEARLINE file can have their latency improved to online latency automatically by staging the file to online cache.
  - OFFLINE files need a human to be involved to achieve online latency.
  - For the SRM we only keep ONLINE and NEARLINE.

```
typedef     struct {
                TRetentionPolicy        retentionPolicy,
                TAccessLatency          accessLatency
            } TRetentionPolicyInfo
```

enum **TAccessPattern** { TransferMode, ProcessingMode }

- TAccessPattern is will be passed as an input to the srmPrepareToGet function. It will make a hint to SRM how the Transfer URL produced by SRM is going to be used. This will not be provided to the srmPrepareToPut because the prepared TURL will always be used for file transfers by wring date into the TURL. If the parameter value is “ProcessingMode”, the system might expect that application will perform some processing of the partially read data, followed by more partial reads and a frequent use of the protocol specific “seek” operation. This will allow optimizations by allocating files on disks with small buffer sizes. If the value is “TransferMode” the file will be read at the highest speed allowed by the connection between the server and a client.
- This may not be sufficient to cover different access patterns as an enumeration. The access pattern may change over time during the life time of files, too. It would be good to have it as a literal strings that a community can “define”.

typedef string **TTransferProtocolInfo**

- TTransferProtocolInfo is an additional information about the transfer protocol when Transfer URL is ready.
- It may specify access speed, available number of parallelism, and other transfer protocol properties.
- Recommended format would be pre-defined XML style such as <parallelism value=2/>, <access speed=40/> etc.

```
typedef     struct {TSURLInfo                fromSURLInfo,
                    TLifeTimeInSeconds       lifetime, // pin time
                    TFileStorageType         preferredFileStorageType,
                    TSpaceToken              targetSpaceToken,
                    TDirOption               dirOption,
                    TAccessPattern           accessPattern
                } TGetFileRequest
```

Note:

- TGetFileRequest includes TAccessPattern which may conflict with the online disk type of the target space associated with target space token if provided. In this case, TAccessPattern must be ignored.

```
typedef     struct {TSURLInfo                toSURLInfo,   // local to SRM
                    TLifeTimeInSeconds       lifetime,     // pin time
                    TFileStorageType         preferredFileStorageType,
                    TRetentionPolicyInfo     targetRetentionPolicyInfo,
                    TSpaceToken              targetSpaceToken,
                    TSizeInBytes             knownSizeOfThisFile
                } TPutFileRequest
```

Note:

- TPutFileRequest does not include TAccessPattern because the prepared TURL has to be for the file transfers, not for local access for analysis.
- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.

```
typedef     struct {TSURLInfo                fromSURLInfo,
                    TSURLInfo                toSURLInfo,
                    TLifeTimeInSeconds       lifetime, // pin time
                    TFileStorageType         targetFileStorageType,
                    TAccessPattern           targetAccessPattern
                    TRetentionPolicyInfo     targetRetentionPolicyInfo,
                    TSpaceToken              targetSpaceToken,
                    TOverwriteMode           overwriteMode,
                    TDirOption               dirOption
                } TCopyFileRequest
```

Note:

- TCopyFileRequest includes TAccessPattern which may conflict with the online disk type of the target space associated with target space token, or target retention policy info, if provided. In this case, TAccessPattern must be ignored.

- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.

```
typedef     struct {TSURL                   fromSURLInfo,
                    TSizeInBytes            fileSize,
                    TReturnStatus           status,
                    TLifeTimeInSeconds      estimatedWaitTimeOnQueue,
                    TLifeTimeInSeconds      estimatedProcessingTime,
                    TTURL                   transferURL
                    TLifeTimeInSeconds      remainingPinTime,
                    TSpaceToken             spaceToken,
                    TTransferProtocolInfo   transferProtocolInfo
            } TGetRequestFileStatus

typedef     struct { TSizeInBytes           fileSize,
                    TReturnStatus           status,
                    TLifeTimeInSeconds      estimatedWaitTimeOnQueue,
                    TLifeTimeInSeconds      estimatedProcessingTime,
                    TTURL                   transferURL,
                    TSURL                   siteURL,  // for future reference
                    TLifeTimeInSeconds      remainingPinTime
                    TSpaceToken             targetSpaceToken,
                    TRetentionPolicyInfo    targetRetentionPolicyInfo,
                    TTransferProtocolInfo   transferProtocolInfo
            } TPutRequestFileStatus

typedef     struct {TSURL                   fromSURL,
                    TSURL                   toSURL,
                    TSizeInBytes            fileSize,
                    TReturnStatus           status,
                    TLifeTimeInSeconds      estimatedWaitTimeOnQueue,
                    TLifeTimeInSeconds      estimatedProcessingTime,
                    TLifeTimeInSeconds      remainingPinTime
                    TSpaceToken             targetSpaceToken,
                    TRetentionPolicyInfo    targetRetentionPolicyInfo,
            } TCopyRequestFileStatus

typedef     struct {TSURL                   fromSURLInfo,
                    TSizeInBytes            fileSize,
                    TReturnStatus           status,
                    TLifeTimeInSeconds      estimatedWaitTimeOnQueue,
                    TLifeTimeInSeconds      estimatedProcessingTime,
                    TLifeTimeInSeconds      remainingPinTime,
                    TSpaceToken             spaceToken,
            } TBringOnlineRequestFileStatus
```

```
typedef     struct {string                  path,   // both dir and file
                    TReturnStatus           status,
                    TSizeInBytes            size,    // 0 if dir
                    TOwnerPermission        ownerPermission,
                    TUserPermission[]       userPermission,
                    TGroupPermission[]      groupPermission,
                    TOtherPermission        otherPermission
                    TGMTTime                createdAtTime,
                    TGMTTime                lastModificationTime,
                    TUserID                 owner,
                    TFileStorageType        fileStorageType,
                    TRetentionPolicyInfo    retentionPolicyInfo,
                    TFileType               type,    // Directory or File
                    TLifeTimeInSeconds      lifetimeAssigned,
                    TLifeTimeInSeconds      lifetimeLeft, // on the SURL
                    TCheckSumType           checkSumType,
                    TCheckSumValue          checkSumValue,
                    TSURL                   originalSURL,   // if path is a file
                    TMetaDataPathDetail[]   subPath          // optional recursive
            } TMetaDataPathDetail
```

```
typedef     struct {TSpaceType              type,
                    TSpaceToken             spaceToken,
                    TRetentionPolicyInfo    retentionPolicyInfo,
                    Boolean                 isValid,
                    TUserID                 owner,
                    TSizeInBytes            totalSize,         // best effort
                    TSizeInBytes            guaranteedSize,
                    TSizeInBytes            unusedSize,
                    TLifeTimeInSeconds      lifetimeAssigned,
                    TLifeTimeInSeconds      lifetimeLeft
            } TMetaDataSpace
```

**srmReserveSpace**

```
    In:     TUserID                 authorizationID,
            TSpaceType              typeOfSpace,
            TRetentionPolicyInfo    preferredRetentionPolicyInfo,
            String                  userSpaceTokenDescription,
            TSizeInBytes            sizeOfTotalSpaceDesired,
            TSizeInBytes            sizeOfGuaranteedSpaceDesired,
            TLifeTimeInSeconds      lifetimeOfSpaceToReserve,
            Int []                  expectedFileSize,
            TStorageSystemInfo      storageSystemInfo
```

| | | |
|---|---|---|
| Out: | TRequestToken | requestToken |
| | TLifeTimeInSeconds | estimatedProcessingTime, |
| | TSpaceType | typeOfReservedSpace, |
| | TRetentionPolicyInfo | retentionPolicyInfo, |
| | TSizeInBytes | sizeOfTotalReservedSpace, // best effort |
| | TSizeInBytes | sizeOfGuaranteedReservedSpace, |
| | TLifeTimeInSeconds | lifetimeOfReservedSpace, |
| | TSpaceToken, | referenceHandleOfReservedSpace, |
| | TReturnStatus | returnStatus |

**notes:**

- Asynchronous space reservation may be necessary for some SRMs to serve many concurrent requests. In such case, request token can be provided. If space reservation can be done immediately, request token must not be returned.
- When asynchronous space reservation is necessary, the returned status code should be SRM_REQUEST_QUEUED or SRM_REQUEST_INPROGRESS.
- *estimateProcessingTime* to complete the space reservation request is returned when known.
- *expectedFileSize* is a hint that SRM server can use to reserve consecutive storage sizes for the request. At the time of space reservation, if space accounting is done only at the level of the total size, this hint wouldn't do any good, and at the time of PrepareToPut, it matters. However, for some SRMs, those hints will make a contribution for a decition to allocate those blocks in some specific disks.
- *lifetimeOfSpaceToReserve is not needed if requesting permanent space.*
- *SRM can provide default size and lifetime if not supplied.*
- *storageSystemInfo is optional in case storage system requires additional security check.*
- *If sizeOfTotalSpaceDesired is not specified, the SRM will return its default quota.*
- *sizeOfTotalReservedSpace* is in best effort bases. For guaranteed space size, *sizeOfGuaranteedReservedSpace* should be checked. These two numbers may match, depending on the storage systems.

**srmGetStatusOfReserveSpace**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TRequestToken | requestToken |

| | | |
|---|---|---|
| Out: | TLifeTimeInSeconds | estimatedProcessingTime, |
| | TRetentionPolicyInfo | retentionPolicyInfo, |
| | TSizeInBytes | sizeOfTotalReservedSpace, // best effort |
| | TSizeInBytes | sizeOfGuaranteedReservedSpace, |
| | TLifeTimeInSeconds | lifetimeOfReservedSpace, |
| | TSpaceToken, | referenceHandleOfReservedSpace, |
| | TReturnStatus | returnStatus |

**notes:**

- If the space reservation is not completed, *estimateProcessingTime* is returned when known. The returned status code in such case should be SRM_REQUEST_QUEUED or SRM_REQUEST_INPROGRESS
- *sizeOfTotalReservedSpace* is in best effort bases. For guaranteed space size, *sizeOfGuaranteedReservedSpace* should be checked. These two numbers may match, depending on the storage systems.

**srmChangeFileStorageType**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TSURLInfo[] | arrayOfPath, |
| | TFileStorageType | desiredStorageType, |
| | TSpaceToken | targetSpaceToken |
| | TRetentionPolicyInfo | targetRetentionPolicyInfo |
| Out: | TReturnStatus | returnStatus, |
| | TSURLReturnStatus[] | arrayOfFileStatus |

**notes:**

- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- *Applies to both dir and file*
- *Either path must be supplied.*
- *If a path is pointing to a directory, then the effect is recursive for all the files in this directory.*
- *Space allocation and de-allocation maybe involved.*

**srmChangeRetentionPolicy**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TSpaceToken | sourceSpaceToken |
| | TRetentionPolicyInfo | sourceRetentionPolicyInfo |
| | TSpaceToken | targetSpaceToken |
| | TRetentionPolicyInfo | targetRetentionPolicyInfo |
| Out: | TReturnStatus | returnStatus, |
| | TSURLReturnStatus[] | arrayOfFileStatus |

**notes:**

- All files in the *sourceSpaceToken* will be moved to the *targetSpaceToken*. Or all files in the *sourceRetentionPolicyInfo* will be moved to the *targetRetentionPolicyInfo*.
- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- Space allocation and de-allocation maybe involved.

**srmRm**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TSURLInfo[] | arrayOfFilePaths |
| | TSpaceToken | spaceToken |
| | TRetentionPolicyInfo | retentionPolicyInfo |
| Out: | TReturnStatus | returnStatus, |
| | TSURLReturnStatus[] | arrayOfFileStatus |

**notes:**

- The need for those additional parameters is because we now need to know which copy of the file needs to be removed. This differentiates from srmRemoveFiles which requires *requestToken* from *srmPrepareToGet* or *srmPrepareToPut.*.
- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- *Applies to files*
- *To distinguish from srmRmDir(), this function applies to files only.*

**srmLs**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TSURLInfo[] | path, |
| | TFileStorageType | fileStorageType, |
| | TSpaceToken | spaceToken |
| | TRetentionPolicyInfo | retentionPolicyInfo, |
| | boolean | fullDetailedList, |
| | boolean | allLevelRecursive, |
| | int | numOfLevels, |
| | int | offset, |
| | int | count |
| Out: | TMetaDataPathDetail[] | details, |
| | TReturnStatus | returnStatus |

**notes:**

- When spaceToken is provided, SRM returns all files that belong to the space associated with the space token. Same for retentionPolicyInfo.
- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- *Applies to both dir and file*
- *fullDetailedList=false by default.*
    - *For directories, only path is required to be returned.*
    - *For files, path and size are required to be returned.*

- *If fullDetailedList=true, the full details are returned.*
  - *For directories, path and userPermission are required to be returned.*
  - *For files, path, size, userPermission, lastModificationTime, typeOfThisFile, and lifetimeLeft are required to be returned, similar to unix command ls –l.*
- *If allLevelRecursive=true then file lists of all level below current will be provided as well.*
- *If allLevelRecursive is "true" it dominates, i.e. ignore numOfLevels. If allLevelRecursive is "false" or missing, then do numOfLevels. If numOfLevels is "0" (zero) or missing, assume a single level. If both allLevelRecursive and numOfLevels are missing, assume a single level.*
- *When listing for a particular type specified by "fileStorageType", only the files with that type will be in the output.*
- *Empty directories will be returned.*
- *We recommend width first in the listing.*
- *We recommend that list of directories come before list of files in the return array (details).*

**srmMv**

| | | | |
|---|---|---|---|
| In: | TUserID | authorizationID, |
| | TSURLInfo | fromPath, |
| | TSURLInfo | toPath |
| Out: | TReturnStatus | returnStatus |

**notes:**

- Now it may involve associated space and retention policy (storage class) changes internally.
- *Applies to both dir and file*
- *Authorization checks need to be performed on both fromPath and toPath.*

**srmBringOnline**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TGetFileRequest[] | arrayOfFileRequest, |
| | string[] | arrayOfTransferProtocols, |
| | string | userRequestDescription, |
| | TStorageSystemInfo | storageSystemInfo, |
| | TLifeTimeInSeconds | totalRetryTime |
| | TSpaceToken | targetSpaceToken |
| | TAccessPattern | accessPattern |
| Out: | TRequestToken | requestToken, |
| | TReturnStatus | returnStatus, |
| | string[] | supportedTransferProtocols, |

TBringOnlineRequestFileStatus[] arrayOfFileStatus

**notes:**

- TAccessPattern may conflict with the online disk type of the target space associated with target space token, if both provided. In this case, TAccessPattern must be ignored.
- When *arrayOfTransferProtocols* are submitted, SRM returns those transfer protocols that SRM supports among the user-submitted transfer protocols.
- *The userRequestDescription is a user designated name for the request. It can be used in the getRequestID method to get back the system assigned request ID.*
- *SRM assigns the requestToken at this time.*
- *"retryTime" means: if all the file transfer for this request are complete, then try previously failed transfers for a total time period of "retryTime".*
- *In case that the retries fail, the return should include an explanation of why the retries failed.*
- *This call is an asynchronous (non-blocking) call. To get subsequent status and results, separate calls should be made.*

**srmStatusOfBringOnlineRequest**

| | | | |
|---|---|---|---|
| In: | TRequestToken | requestToken, |
| | TUserID | authorizationID |
| | TSURL[] | arrayOfFromSURLs, |
| Out: | TReturnStatus | returnStatus, |
| | TBringOnlineRequestFileStatus[] | arrayOfFileStatus |

**notes:**

- *If* arrayOfFromSURLs *is not provided, returns status for all files in this request.*

**srmPrepareToGet**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TGetFileRequest[] | arrayOfFileRequest, |
| | string[] | arrayOfTransferProtocols, |
| | string | userRequestDescription, |
| | TStorageSystemInfo | storageSystemInfo, |
| | TLifeTimeInSeconds | totalRetryTime |
| | TSpaceToken | targetSpaceToken |
| | TAccessPattern | accessPattern |
| | TTransferProtocolInfo | transferProtocolInfo |
| Out: | TRequestToken | requestToken, |
| | TReturnStatus | returnStatus, |
| | TGetRequestFileStatus[] | arrayOfFileStatus |

**notes:**

- If TAccessPattern is provided at the request-level and file-level, then file-level TAccessPattern will take the priority and must be used.

- If TSpaceToken is provided at the request-level and file-level, then file-level TSpaceToken will take the priority and must be used.
- If TTransferProtocolInfo is provided at the request-level and file-level, then file-level TTransferProtocolInfo will take the priority and must be used.
- The *userRequestDescription* is a user designated name for the request. It can be used in the *getRequestID* method to get back the system assigned request ID.
- Only pull mode is supported for file transfers that client must pull the files from the TURL.
- SRM assigns the *requestToken* at this time for asynchronous status checking.
- Normally this call will be followed by *srmRelease*().
- "*retryTime*" means: if all the file transfer for this request are complete, then try previously failed transfers for a total time period of "*retryTime*".
- In case that the retries fail, the return should include an explanation of why the retries failed.
- This call is an asynchronous (non-blocking) call. To get subsequent status and results, separate calls should be made.
- When the file is ready for the user, the file is implicitly pinned in the cache and lifetime will be enforced.
- The invocation of *srmReleaseFile*() is expected for finished files later on.

**srmPrepareToPut**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TPutFileRequest[] | arrayOfFileRequest, |
| | string[] | arrayOfTransferProtocols, |
| | string | userRequestDescription, |
| | TOverwriteMode | overwriteOption, |
| | TStorageSystemInfo | storageSystemInfo, |
| | TLifeTimeInSeconds | totalRetryTime |
| | TRetentionPolicyInfo | targetRetentionPolicyInfo |
| | TSpaceToken | targetSpaceToken |
| | TTransferProtocolInfo | transferProtocolInfo |
| Out: | TRequestToken | requestToken, |
| | TReturnStatus | returnStatus, |
| | TPutRequestFileStatus[] | arrayOfFileStatus |

**notes:**

- If TSpaceToken is provided at the request-level and file-level, then file-level TSpaceToken will take the priority and must be used.
- If TRetentionPolicyInfo is provided at the request-level and file-level, then file-level TRetentionPolicyInfo will take the priority and must be used.
- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- If TTransferProtocolInfo is provided at the request-level and file-level, then file-level TTransferProtocolInfo will take the priority and must be used.

- Only push mode is supported for file transfers that client must push the file to the prepared TURL.
- *StFN* ("*toSURLInfo*" in the *TPutFileRequest*) has to be local to SRM. If *stFN* is not specified, SRM will name it automatically and put it in the specified user space. This will be returned as part of the "Transfer URL".
- *srmPutDone*() is expected after each file is "put" into the allocated space.
- The lifetime of the file starts as soon as SRM get the *srmPutDone*(). If *srmPutDone*() is not provided then the files in that space are subject to removal when the space lifetime expires.
- "*retryTime*" is meaningful here only when the file destination is not a local disk, such as tape or MSS.
- In case that the retries fail, the return should include an explanation of why the retries failed.

**srmCopy**

| | | |
|---|---|---|
| In: | TUserID | authorizationID, |
| | TCopyFileRequest[] | arrayOfFileRequest, |
| | string | userRequestDescription, |
| | TOverwriteMode | overwriteOption, |
| | Boolean | removeSourceFiles (default = false), |
| | TStorageSystemInfo | storageSystemInfo, |
| | TLifeTimeInSeconds | totalRetryTime |
| | TRetentionPolicyInfo | targetRetentionPolicyInfo |
| | TSpaceToken | targetSpaceToken |
| Out: | TRequestToken | requestToken, |
| | TReturnStatus | returnStatus, |
| | TCopyRequestFileStatus[] | arrayOfFileStatus |

**notes:**

- If TSpaceToken is provided at the request-level and file-level, then file-level TSpaceToken will take the priority and must be used.
- If TRetentionPolicyInfo is provided at the request-level and file-level, then file-level TRetentionPolicyInfo will take the priority and must be used.
- If TSpaceToken AND TRetentionPolicyInfo are provided, then their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- *Pull mode: copy from remote location to SRM. (e.g. from remote to MSS.)*
- *Push mode: copy from SRM to remote location.*
- *Always release files from source after copy is done.*
- *When removeSourceFiles=true, then SRM will remove the source files on behalf of the caller after copy is done.*
- *In pull mode, send srmRelease() to remote location when transfer is done.*

- *If in push mode, then after transfer is done, notify the caller. User can then release the file. If user releases a file being copied to another location before it is done, then refuse to release.*
- *Note there is no protocol negotiation with the client for this request.*
- *"retryTime" means: if all the file transfer for this request are complete, then try previously failed transfers for a total time period of "retryTime".*
- *In case that the retries fail, the return should include an explanation of why the retires failed.*
- *When both fromSURL and toSURL are local, perform local copy*
- *Empty directories are copied as well.*

**srmExtendFileLifeTimeInSpace**

| | | |
|---|---|---|
| In: | TSpaceToken | spaceToken, |
| | TRetentionPolicyInfo | retentionPolicyInfo |
| | TSURL | siteURL, |
| | TUserID | authorizationID, |
| | TLifeTimeInSeconds | newLifeTime |
| Out: | TReturnStatus | returnStatus, |
| | TLifeTimeInSeconds | newTimeExtended |

**notes:**

- When *spaceToken* is provided, the lifetime of the file copy of the SURL in the space associated with the space token will be extended.
- When retention policy info is provided, the lifetime of the file copy of the SURL associated with the retention policy info will be extended .
- If retention policy info is provided AND spaceToken is provided, their types must match exactly. Otherwise, the request must be rejected. From the client's point of view, it would be better to use only one of these parameters.
- *newLifeTime is relative to the calling time. Lifetime will be set from the calling time for the specified period.*
- *The number of lifetime extensions maybe limited by SRM according to its policies.*
- *If original lifetime is longer than the requested one, then the requested one will be assigned.*
- *If newLifeTime is not specified, the SRM can use its default to assign the newLifeTime.*

**enum TStorageAttributes** { SRM_FILE_STORAGE_TYPE,
SRM_STORAGE_CAPACITY,
SRM_USER_STORAGE_MAX,
SRM_USER_STORAGE_MIN,
SRM_USER_STORAGE_DEFAULT_LIFETIME,
SRM_DEFAULT_FILE_LIFETIME,
SRM_DEFAULT_FILE_STORAGE_TYPE,
SRM_DEFAULT_TURL_EXPIRATION_TIME,

SRM_ DEFAULT_ACCESS_LATENCY,
SRM_ DEFAULT_ACCESS_PATTERN,
SRM_ DEFAULT_RETENTION_POLICY,
SRM_SUPPORTED_RETENTION_POLICY,
SRM_SUPPORTED_ACCESS_PATTERN,
SRM_SUPPORTED_ACCESS_LATENCY,
SRM_DEFAULT_TRANSFER_PROTOCOL
}

Note:

- This is to discover what features an SRM supports and what the default values for a specific features in SRM.
- We can add more here.

```
typedef         struct   {
                         TStorageAttributes    storageAttr,
                         string                value,
                         string                valueType,
                         string                explanation
                         } TStorageInfo
```

Note:

- value - value of the *storageAttr*. When SRM supports multiple values (e.g. when SRM supports multiple retention policies), this value may contain more than one value separated by comma (,). E.g. RELICA,CUSTODIAL
- valueType - data type of the value for storageAttr in literal characters. For example, int, long, string, boolean, TRetentionPolicy, etc.
- explanation – this parameter explains what the value means to the SRM server. E.g. CUSTODIAL from TRetentionPolicy can be explained in the parameter that how the particular SRM treats this type if supported.

**srmGetSRMStorageInfo**

```
        In:     TUserID                 authorizationID,
                EnumStorageAttributes   desiredAttributes[]

        Out:    TReturnStatus           returnStatus,
                TStorageInfo            storageInfo[]
```

**notes:**

- *srmGetSRMStorageInfo* retrieves SRM storage information, such as storage capacity, client quota, default lifetime, etc.
- When output parameter, TStorageInfo is returned to the client, *storageAttr* and its *value* are required to be returned.

```
typedef         struct   {
```

string transferProtocol,
string attributes
} **TTransferProtocols**

Note:

- string *transferProtocol* (required) : Supported transfer protocol. For example, gsiftp, http
- string *attributes :* Informational hints for the paired transfer protocol, such how many number of parallel streams can be used, desired buffer size, etc. Recommended to use the key/value pairs as a string list, separated by comma (,) (e.g. buffer_size=1000,parallel=4) or an XML form (e.g. <parallelism value=2/>, <access speed=40/>).

**srmGetTransferProtocols**

| | | | |
|---|---|---|---|
| In: | TUserID | authorizationID, | |
| Out: | TReturnStatus | returnStatus, | |
| | TTransferProtocols | protocolInfo[] | |

**notes:**

- *srmGetTransferProtocols*() returns the supported file transfer protocols in the SRM with any additional information about the transfer protocol.